Induction and Co-induction in Sequent Calculus

Alberto Momigliano
Department of Mathematics and Computer Science
University of Leicester, U

Joint work with Dale Miller and Alwen Tiu

Types 2003. Turin, May 1st, 2003

Overview of the Talk

- Motivations and history.
- Partial Inductive Definitions and $FO\lambda^{\Delta\mathbb{N}}$.
- Induction and Coinduction rules.
- Implementation
- Circular proofs
- Future work

Motivations

- Using logical framework to reason about computation systems.

- Deductive systems (e.g., operational semantics) encoded as logical theories. Properties of the encoded system can be verified inside the logical framework possibly via proof search paradigm.

- Support **Higher Order Abstract Syntax** (HOAS).

- To preserve adequacy of encodings, framework is intentionally *weak*, but must allow support for (co)inductive reasoning.

Relevant History (Sketch)

- From Gentzen to iterated inductive defs [Martin-Löf'71]
- Impredicative encoding of inductive types in system F [Böhm & Berarducci'85].
- Mendler-style (co)inductive types [Mendler'87], [Geuvers'92]
- CC^{Ind} [Pfenning & Paulin '90], CC^{CoInd} [Gimenez'96]
- Work on type-based termination [Barthe et al'03], (co)iteration [Abel et al'02], category-theoretic analysis [Uustalu et al'00] etc...
- Sequent calculus: not much.

- Sequent calculus for intuitionistic logic $\Gamma \longrightarrow C$ over simply typed λ-terms, but no quantification of predicates.

- Left and right rules for introducing defined predicates with possibly non-Horn clauses [Halnass'91, S. Heister'93].

- Differently from set(type)-theoretic inductive definitions, case analysis (left rule) is not bungled up with induction principles and monotonicity.

- A **definition** is a finite set of *clause*

$$\forall \bar{x}[p_1(\bar{t}_1) \triangleq B_1] \quad \ldots \quad \forall \bar{x}[p_n(\bar{t}_n) \triangleq B_n] \quad (n \geq 0)$$

  The symbol $\triangleq$ not a logical connective, just to denote definitional clauses.

Partial Inductive Definitions

- Fix a definition and a equality theory, here ho-patterns.

- Right rule: as a logic program, this is *backchaining*.

$$\frac{\Gamma \longrightarrow B\theta}{\Gamma \longrightarrow A}\,(\mathit{def}\mathcal{R}), H\theta = A, \text{ for some clause } \forall\bar{x}[H \stackrel{\triangle}{=} B]$$

- Left rule:

$$\frac{\{B\theta, \Gamma\theta \longrightarrow C\theta \mid \ \theta A = \theta H, \text{ for some clause } \forall\bar{x}[H \stackrel{\triangle}{=} B]\}}{A, \Gamma \longrightarrow C}\,(\mathit{def}\mathcal{L})$$

- *Unification* used to select clauses from a definition. Eigenvariables can be instantiated. The set of premises can be empty, finite, or infinite since in some theories CSUs are not effectively computable.

Ex: Big Step Lazy Evaluation

$$\textit{Terms} \quad e \quad ::= \quad x \mid \mathbf{fun}\ x\,.\ e \mid e \ @\ e' \mid \mathbf{fix}\ x.\ e$$

- Encode it in Full HOAS (no separate category of variables): object level binder represented via meta-level binder λ from the logic, for example $\ulcorner \mathbf{fix}\ \ x.\ x\ @\ x \urcorner = \mathtt{fix}(\lambda \mathtt{x}\ .\ \mathtt{x}\ @\ \mathtt{x})$:

```
exp  : type.
fun  : (exp -> exp) -> exp.
@    : exp -> exp -> exp.
fix  : (exp -> exp) -> exp.
```

- Note: `exp` **cannot** be a data-type because of negative occurrences, so those are simply constants. Crucially, **freeness** properties must be derivable, wrt inversion on judgments or case analysis on syntax.

Free Equality

- There are choices. Standard one (discarded here to keep linearity of heads) is equality as a definition $X = X \stackrel{\triangle}{=} \top$. Freeness follow by *def$\mathcal{L}$*, see injectivity of OL binders:

$$\frac{\top \longrightarrow E = E}{\mathbf{fun}\ x\,.\ E\ x = \mathbf{fun}\ x\,.\ F\ x \longrightarrow E = F}\ \mathit{def}\mathcal{L}$$

- Sequent calculus rendition of free equality [S-Heister'94], ex:

$$\frac{f \not\equiv g}{\Gamma, f\ \vec{s} = g\ \vec{t} \longrightarrow C}\ Clsh \qquad \frac{\Gamma[t/x] \longrightarrow A[t/x]}{\Gamma, t = x \longrightarrow A}\ Rpl$$

- Exploiting unification, which is now restricted to equality reasoning (definitions as Clark's completion). We adopt it.

$$\frac{\{\Gamma\theta \longrightarrow C\theta \mid \theta \in CSU(s,t)\}}{\Gamma, s = t \longrightarrow C}\ \mathit{eq}\mathcal{L} \qquad \frac{s =_{\beta\eta} t}{\Gamma \longrightarrow s = t}\ \mathit{eq}\mathcal{R}$$

$$\frac{e_1 \Downarrow \mathbf{fun}\ x\,.\ e_1' \qquad [e_2/x]e_1' \Downarrow v}{e_1\ @\ e_2 \Downarrow v}\ \mathbf{ev\_app}$$

$$\frac{}{\mathbf{fun}\ x\,.\ e \Downarrow \mathbf{fun}\ x\,.\ e}\ \mathbf{ev\_fun} \qquad \frac{[\mathbf{fix}\ x.\ e/x]e \Downarrow v}{\mathbf{fix}\ x.\ e \Downarrow v}\ \mathbf{ev\_fix}$$

- This corresponds to the following PID (note HOAS):

$$\begin{aligned}
\mathbf{fun}\ E \Downarrow \mathbf{fun}\ E &\stackrel{\triangle}{=} \top \\
(E_1\ @\ E_2) \Downarrow V &\stackrel{\triangle}{=} \exists E' : E_1 \Downarrow \mathbf{fun}\ E' \wedge (E'\ E_2) \Downarrow V \\
\mathbf{fix}\ E \Downarrow V &\stackrel{\triangle}{=} E\ (\mathbf{fix}\ E) \Downarrow V
\end{aligned}$$

- *def$\mathcal{R}$* provides Prolog-like computations, *def$\mathcal{L}$* case analysis:

$$\frac{\top \longrightarrow \mathbf{fun}\ E = \mathbf{fun}\ E}{\mathbf{fun}\ E \Downarrow V \longrightarrow V = \mathbf{fun}\ E}\ \mathit{def}\mathcal{L}$$

(Co)Induction

- To prove anything interesting, need some form of induction. In [McDowell'97], only induction on natural numbers:

$$\frac{}{\Gamma \longrightarrow \mathit{nat}\; z}\, \mathit{nat}\mathcal{R} \qquad\qquad \frac{\Gamma \longrightarrow \mathit{nat}\; I}{\Gamma \longrightarrow \mathit{nat}\; s\; I}\, \mathit{nat}\mathcal{R}$$

$$\frac{\longrightarrow D\, z \quad D\, j \longrightarrow D\,(s\; j) \quad D\, I, \Gamma \longrightarrow C}{\mathit{nat}\; I, \Gamma \longrightarrow C}\, \mathit{nat}\mathcal{L}$$

- Introduce counters in definitions, derive and use complete induction (doable but clunky):

$$\mathbf{fix}\;\; E \Downarrow^{(s\; N)} V \;\; \stackrel{\triangle}{=} \;\; E\;\; (\mathbf{fix}\;\; E) \Downarrow^{N} V \ldots$$

- But what about coinductive predicates? Inductive encoding works only with co-continuity, need direct approach.

$$\begin{aligned} (E_1 @ E_2) \Uparrow \;\; &\stackrel{\triangle}{=} \;\; E_1 \Uparrow \vee \exists E' : E_1 \Downarrow \mathbf{fun}\;\; E' \wedge (E'\;\; E_2) \Uparrow \\ \mathbf{fix}\;\; E \Uparrow \;\; &\stackrel{\triangle}{=} \;\; E\;\; (\mathbf{fix}\;\; E) \Uparrow \end{aligned}$$

- For convenience, we define predicate with a single (disjunctive) body, where pattern matching is a constraint ex: $\mathit{nat}\, x$ has body $(x = \mathsf{z}) \vee \exists y.(x = \mathsf{s}\, y) \wedge \mathit{nat}\, y$. For this talk, assume also no mutual recursion.

- **Inductive** predicates $\forall \bar{x}. p\ \bar{x} \stackrel{\mu}{=} B$, holding in every **least** fixed point:

$$\frac{I\ \bar{t}, \Gamma \longrightarrow C \qquad\qquad B_I\ \bar{x} \longrightarrow I\ \bar{x}}{p\ \bar{t}, \Gamma \longrightarrow C}\,\mathcal{IL} \qquad\qquad \frac{\Gamma \longrightarrow B\ \bar{t}}{\Gamma \longrightarrow p\ \bar{t}}\,\mathcal{IR}$$

  where I is the **Invariant** predicate, $\bar{x}$ fresh eigenvariables and B_I is B where every occurrence of p is replaced by I.

- Note: the old $\mathit{nat}\mathcal{L}$ rule is now derivable via the invariant $I = \lambda x.[(x = \mathsf{z}) \vee \exists y.(x = \mathsf{s}\, y)] \wedge D\ x$. In general, $\mathit{def}\mathcal{L}$ is derivable via $\mathcal{IL}$.

- Easiest example in the book: value soundness $e \Downarrow v \longrightarrow value\ v$. Let $I = \lambda xy\ .\ value\ y$.

$$\frac{I\ e\ v \longrightarrow C \qquad\qquad B_I\ e\ v \longrightarrow I\ e\ v}{e \Downarrow v \longrightarrow C}\,\mathcal{IL}$$

- Invariant implies conclusion $value\ v \longrightarrow value\ v$.

- Inductive step(s):

1. $x = \mathbf{fun}\ E = y \longrightarrow value\ y$. $indR$ on $value(\mathbf{fun}\ E)$.

2. $x = E_1 \mathbin{@} E_2 \wedge y = V \wedge \ldots I\ (E'\ E_2)\ V \longrightarrow value\ y$

3. $x = \mathbf{fix}\ F \wedge y = V \wedge I\ (F\ (\mathbf{fix}\ F))\ V \longrightarrow value\ y$.

- **Conductive** predicates $\forall \bar{x}.p\ \bar{x} \stackrel{\nu}{=} B$, holding in every **greatest** fixed point. S is the purported **Simulation**:

$$\frac{B\ \bar{t}, \Gamma \longrightarrow C}{p\ \bar{t}, \Gamma \longrightarrow C}\mathcal{CIL} \qquad \frac{\Gamma \longrightarrow S\ \bar{t} \quad S\ \bar{x} \longrightarrow B_S\ \bar{x}}{\Gamma \longrightarrow p\ \bar{t}}\mathcal{CIR}$$

- Proviso lvl(S) $\leq$ lvl(p). Note: the *def$\mathcal{R}$* rule is derivable.

- Example: let $h \equiv \mathbf{fix}\ (\lambda x.\ x \ @\ x)$. It holds $\longrightarrow h \Uparrow$. Let S be $\lambda y.\ y = h \vee y = h \ @\ h$. LHS is
$\longrightarrow h = h \vee h = h \ @\ h$. RHS yields two subgoals
$\longrightarrow h \ @\ h = E_1 \ @\ E_2 \wedge (E_1 = h \ @\ h \vee E_1 = h)$ and
$\longrightarrow \exists E.\ h = \mathbf{fix}\ E \wedge (E\ (\mathbf{fix}\ E) = h \ @\ h \vee E\ (\mathbf{fix}\ E) = h)$

- **Regular** predicates $\forall \bar{x}.p\ \bar{x} \stackrel{\triangle}{=} B$, true in **every** fixed point

- Without restrictions, PID may be inconsistent – cut not eliminable – even before (co)inductive rules, see $p \stackrel{\triangle}{=} p \supset \perp$.

- Give predicates and formulas a level and require definitions to be stratified. This brings back monotonicity at the level of definitions, but supports HOAS. Need to go two-level to accommodate reasoning with non-stratifiable hypothetical judgments (not unlike $\mathcal{M}_\omega$).

- Use dependency graph to avoid a predicate to be (mutually) defined both inductively and co-inductively.

- Cut-elimination based on adaptation of Tait-style *reducibility*, generalizes McDowell's proof for $FO\lambda^{\Delta\mathbb{N}}$.

- Well-order on proofs induced by "local" cut reduction, proof by induction on the length of proof of the rightmost derivation, and on the reducibility of the left. Coinduction may need notion of "coreducibility".

- Given $A\bar{y} \stackrel{\mu}{=} B[A]\ \bar{y}$

$$\cfrac{\cfrac{\overset{\Pi_1}{\Delta \longrightarrow B[A]\ \bar{t}}}{\Delta \longrightarrow A\ \bar{t}}\,\mathcal{IR} \qquad \cfrac{\overset{\Pi_B}{B[S]\ \bar{y} \longrightarrow S\ \bar{y}} \qquad \overset{\Pi}{S\ \bar{t}, \Gamma \longrightarrow C}}{A\ \bar{t}, \Gamma \longrightarrow C}\,\mathcal{IL}}{\Delta, \Gamma \longrightarrow C}\,mc \quad .$$

- This reduces to

$$\cfrac{\cfrac{\overset{\Pi'_1}{\Delta \longrightarrow B[S]\ \bar{t}} \qquad \overset{\Pi_B[\bar{t}/\bar{y}]}{B[S]\ \bar{t} \longrightarrow S\ \bar{t}}}{\Delta \longrightarrow S\ \bar{t}}\,mc \qquad \overset{\Pi}{S\ \bar{t}, \Gamma \longrightarrow C}}{\Delta, \Gamma \longrightarrow C}\,mc$$

Implementation

- Hybrid [Ambler et al'02], a package on top of Isabelle/HOL.

- Generalizing [A. Gordon'94], a deep encoding in Isabelle/HOL of a λ-calculus which provides a form of *logical framework* where the syntax of an object level logic can be adequately represented by HOAS.

- Supports *tactical theorem proving*, *(co)induction*, but *definitional*, so consistent *within a classical type theory*. This gives access to all the features of Isabelle HOL, in particular (co)induction.

- Bottom line: 'Hybrid syntax' provides the user with a form of HOAS, but this is syntactic sugar (more properly a **definition**) for an underlying de Bruijn representation.

- Since stratified PID are monotonic, implement them as a Isabelle HOL(co)inductive definition (**abstr** annotations to isolate the parametric function space, to be internalized):

$$
\begin{array}{rcl}
\mathit{coinductive}\ \_\Uparrow & :: & \mathit{exp} \Rightarrow \mathit{bool} \\
[\![\ E_1 \Uparrow\ ]\!] & \Longrightarrow & (E_1 @ E_2) \Uparrow \\
[\![\ E_1 \Downarrow \mathbf{fun}\ \ E';\ \ \mathsf{abstr\,arg}\, E';\ \ (E'\ \ E_2) \Uparrow\ ]\!] & \Longrightarrow & (E_1 @ E_2) \Uparrow \ldots
\end{array}
$$

- **Prove** (by comprehension) the set-theoretic soundness of the $\mathcal{CIR}$ rule, by Isabelle HOL's co-induction, where $B_{\Uparrow}$ is the completion body of the definition, abstracted over S:

$$\mathit{Goal}\ ''(\exists S.\ S\ \ t \wedge (\forall y.\ S\ \ y \rightarrow B_{\Uparrow}\ S\ \ y)) \rightarrow t \Uparrow '';$$

- Use the latter to prove: $\mathit{Goal}\ ''\,\mathbf{fix}\ (\lambda x.\ x\ @\ x) \Uparrow '';$. Let S be $\lambda x.\, x = \mathbf{fix}\ (\lambda x.\ x\ @\ x) \vee x = (\mathbf{fix}\ (\lambda x.\ x\ @\ x)\ @\ \mathbf{fix}\ (\lambda x.\ x\ @\ x))$.

- Using $\mathcal{CIR}$ requires the invention of a simulation: An alternative lazy way is to drop the rule and do proof search and circularity checking (guarded induction), as suggested by Coquand and implemented in CC^{CoInd}.

- Derivations (with *def*$\mathcal{L}$*def*$\mathcal{R}$) may include cycles. The derivation of a sequent can be closed if there is a leaf which is an instance of another sequent in the derivation graph.

- Circular branch has to be *guarded*, i.e., there has to be an application of *def*$\mathcal{R}$ in the branch (up to permutation). Equivalent in sequents to *productivity* in type theory.

- Restriction: circularity across cut rule must be restricted. Otherwise we lose cut-elimination.

- Divergence (as before), for $h \equiv \mathbf{fix}\ (\lambda x.\ x \mathbin{@} x)$:

$$\dfrac{\dfrac{\longrightarrow h \Uparrow}{\longrightarrow h \mathbin{@} h \Uparrow}\ \mathit{def}\mathcal{R}}{\longrightarrow h \Uparrow}\ \mathit{def}\mathcal{R}$$

- Applicative simulation [Abramski'90]:

$$R \leq S \ \stackrel{\nu}{=}\ \forall T.\ R \Downarrow \mathbf{fun}\ T \to (\exists U. S \Downarrow \mathbf{fun}\ U \wedge \forall p.\ (T\ \ p) \leq (U\ \ p))$$

- Transitivity $m_1 \leq m_2, m_2 \leq m_3 \longrightarrow m_1 \leq m_3$. Two *def*$\mathcal{L}$ and one of *def*$\mathcal{R}$ and quantifier eliminations (on the right): get $(m_1\ p) \leq (m_2\ p), (m_2\ p) \leq (f_3\ p) \longrightarrow (m_1\ p) \leq (f_3\ p)$ and close.

- $S = \lambda\ xy.\ \exists w\,.\, x \leq w \wedge w \leq y$ constructed via proof-search.

- Cut free circular proofs **not** complete for infinite behavior. For example take *simulation* in CCS:

$$\textit{sim}\ P\ Q \stackrel{\triangle}{=} \forall A \forall P'.P \xrightarrow{A} P' \supset \exists Q'.Q \xrightarrow{A} Q' \wedge \textit{sim}\ P\ '\ Q'$$

- *sim* $P\ Q$, where $P = \mu x.(a.x)$ and $Q = \mu x.((a.x \mid a.x))$.

$$P \xrightarrow{a} P \xrightarrow{a} P \xrightarrow{a} \ldots$$

$$Q \xrightarrow{a} (Q \mid a.Q) \xrightarrow{a} (Q \mid Q) \xrightarrow{a} ((Q \mid a.Q) \mid Q) \xrightarrow{a} \ldots$$

  but there is no Cut free circular proof of it.

- But you can prove it via $\mathcal{CIR}$ with the simulation

$$S := \lambda P \lambda Q.(P = \mu x.a.x) \wedge \exists Q'.Q \xrightarrow{a} Q \mid Q'.$$

Future Work

- Take circular proofs seriously. Show soundness w.r.t. the $\mathcal{CIR}$ rule. Study completeness for certain class of problems (cf simulations). Investigate circular (well-founded) proofs in the **inductive** setting [Sprenger & Dam'03] and the relation with fixed point logics.

- Play a similar game with LF and *Twelf*:

  1. coverage and guardedness checking at the LF level

  2. add ν to the meta-logic $\mathcal{M}_\omega$.

- Look at Mendler-style (co)induction, also to ensure guardedness as-you-go [Gimemenez'98].

Future Work

- Extensions: add the ∇ quantifier [Miller & Tiu'03] and prove consistency of the full system.

- Implementation:

    1. Use Hybrid, which is currently been extended to support ∇ [Ambler et al'03]

    2. A dedicated system for PID's, possibly on top of an interactive proof assistant (Isabelle, Coq).

- Experiment with one big case-study, say congruence of bisimulation in the higher-order π-calculus.